Компьютерная лингвистика в информационном поиске

Лекция 4

БГУ ФПМИ, 2018

План

Исправление опечаток

Синонимия

- Генерация тезауруса

Векторные представления слов

План

10-15% запросов содержат ошибки: пропуск символов, замена символов, слитное написание.
Способы исправления:

- исправление отдельных термов
- исправление с учетом контекста

Базовая схема опечаточника

1. Проверить наличие терма в словаре.
2. Найти ближайший в некотором смысле терм из словаря.
3. Если ближайших несколько – выбрать более частотный.

Отображение для пользователя

- Выдача по исходному запросу с предожением исправить.
- Выдача по исправленному запросу с уведомлением об исправлении.
- Смешанная выдача.

Редакторское расстояние

Расстояние Левенштейна $L(w_1, w_2)$

Это минимальное количество операций вставки одного символа, удаления одного символа и замены одного символа на другой, необходимых для превращения w_1 в w_2 (и наоборот).

Редакторское расстояние

$$M = int[|w_1|, |w_2|]$$

$$M[i,j] = \begin{cases} 0 & , i = j = 0 \\ i & , j = 0, i > 0 \\ j & , i = 0, j > 0 \\ \min\{M[i-1,j] + 1, M[i,j-1] + 1, & \\ \quad M[i-1,j-1] + (w_1[i] != w_2[j])\} & , i,j > 0 \end{cases}$$

$$L(w_1, w_2) = M[|w_1| - 1, |w_2| - 1]$$

Редакторское расстояние: эффективность

Время – $O(|w_1| \times |w_2|)$, память – $O(2 \cdot \min\{|w_1|, |w_2|\})$

Редакторское расстояние: эффективность

Время – $O(|w_1| \times |w_2|)$, память – $O(2 \cdot \min\{|w_1|, |w_2|\})$

- Поиск ближайшего слова по всему словарю слишком затратный.
- Эвристики:
  - перестановки
  - сочетания из букв слова
  - слова с тем же префиксом

Пересечение k-грамм

1. Для словаря сторим k-граммный индекс.
2. Слово запроса разбиваем на k-граммы.
3. Ищем k-граммы в индексе и извлекаем те термы, которые имеют значение меры Жаккара больше порога.

$$J(q,t) = \frac{|\text{kgrams}(q) \cap \text{kgrams}(t)|}{|\text{kgrams}(q) \cup \text{kgrams}(t)|}$$

Soundex

Фонетические исправления – исправление ошибок, возникающих, когда пользователь вводит запрос так, как его слышит.

1. Преобразуем каждый терм словаря в 4-символьный код. Строим инвертированный индекс таких кодов (soundex-index).
2. Преобразовываем термы запроса, ищем их в soundex-индексе.

Soundex: преобразование терма

1. запишем первую букву – её оставляем неизменной
2. осуществим замену символов:
   - b,f,p,v → 1
   - c,g,j,k,q,s,x,z → 2
   - d,t → 3
   - l → 4
   - m,n → 5
   - r → 6
3. все повторы заменяем на 1 символ
4. удалим все символы a,e,h,i,o,u,w,y
5. обрезаем до первых 4 символов (дополняем 0, если нужно).

hermann → h655, pointer → p536.

План

Синонимы

Слова одной части речи, различные по звучанию и написанию, но имеющие близкое лексическое значение. Используются для расширения запросов.

Синонимы: как получить?

- Готовые тезаурусы.
- Данные Википедии (граф объектов).
- Семантические сети (Wordnet).

Составление тезауруса

- Использование контролируемого словаря (отображение терма в каноническое понятие).
- Вручную редакторами.
- Автоматически на основе встречаемости в документах.
- Автоматически на основе пользовательского поведения.

Автоматическая генерация тезауруса

- $PMI(w_1, w_2) = \log \frac{p(w_1)p(w_2)}{p(w_1,w_2)}$
- $cos(V(w_1), V(w_2))$

План

LSA

Пусть C – $m \times n$ матрица терм-документ ранга r.

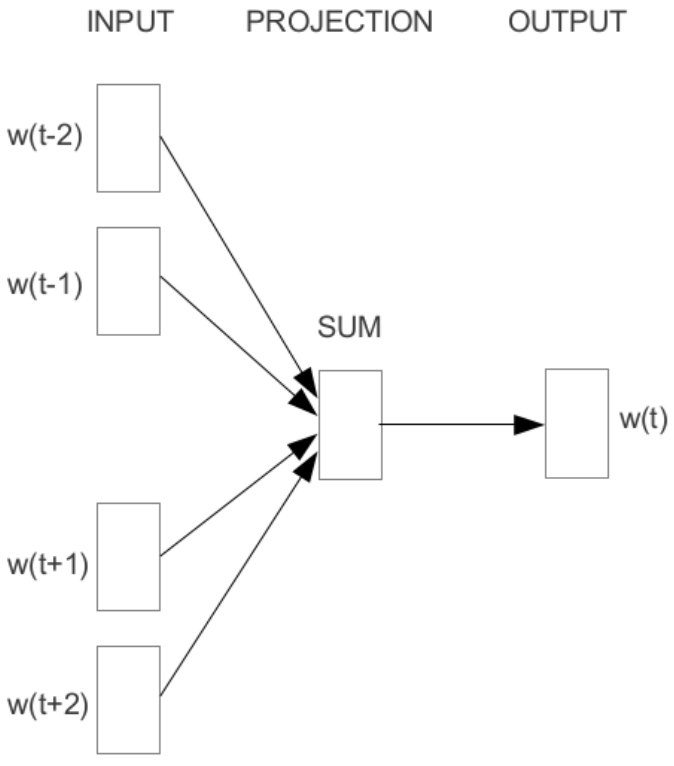
$$C = U\Sigma V^T$$

U – $m \times m$ ортогональная , V – $n \times n$ ортогональная, Σ – $m \times n$ диагональная матрица сингулярных чисел C

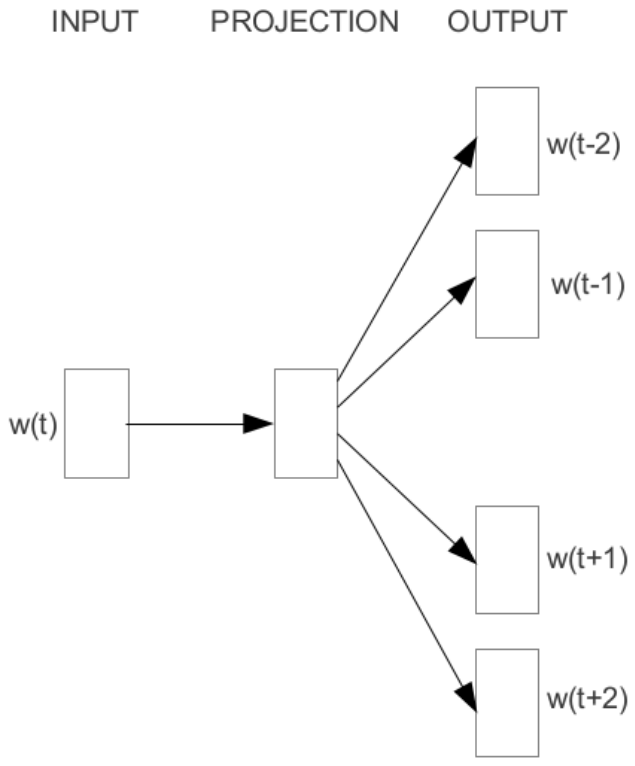
LSA

- Выбираем $k \leq r$ – размерность представления.
- Строки U, обрезанные до длины k, используем как векторные представления слов.

Word2Vec



CBOW



Skip-gram

Word2Vec

- размер внутреннего слоя от 100 до 600
- обучение на $\sim 1B$ слов, 3-50 проходов по коллекции
- рассматривается окружение в 8 слов для CBOW и 20 в Skip-gram
- при обучении skip-gram для каждого w_t сэмплируется слово из его окружения w_{t+j} с вероятностью, обратно пропорциональной $|j|$
- skip-gram модель работает лучше

Применение

- задачи, требующие определения similarity между словами
- классификация текстов
- расширение запроса, автодополнение

Следующая лекция

Поисковый робот